Exercises for the Course

Logic Programming Engineering

(Author Dr. W. Nauber)
Dr P. Bruscoli, Peter Steinke, Amin Timany

Practical session 6

Backtracking I

Exercise 6.1 crypto-arithmetic puzzle

A typical crypto-arithmetic puzzle is shown below:

```
  a b c
+ b c c
  -----
  c a a
  =====
```

where we need to assign decimal digits to letters `a`, `b`, `c` in such a way that the sum holds. Different letters must be assigned different digits.

A possible solution for this problem is to generate, with the help of backtracking, conversion tables (from letters into digits), and then, converting lists of digits into numbers, in order to check the correctness of the arithmetic problem.

a) Define a predicate *cryptadd*($S1$, $S2$, R, $D1$, $D2$, $D3$) which for given lists of letters $S1$, $S2$, and R checks if the corresponding crypto-arithmetic puzzle for addition of two numbers have a solution. An existing solution is given as lists of digits $D1$, $D2$, and $D3$, respectively corresponding to $S1$, $S2$, and R.

Example: $? - cryptadd([a, b, c], [b, c, c], [c, a, a], D1, D2, D3)$. yields
$D1 = [0, 4, 5]$,
$D2 = [4, 5, 5]$, and
$D3 = [5, 0, 0]$.

Hints for realization in Prolog:

Define a predicate *gendigits*(L, DL, UL, UD, NDL, NUL, NUD, D) which for a given list of letters L produces a list of corresponding digits D. In converting a letter we shall consider two cases: the letter occurs for the first time or the letter has been already converted.
In the first case the letter is converted by a non-deterministic selection of a digit from list of digits DL; in the second case the conversion takes place with the help of the conversion table encoded as lists UL (used letters) and UD (used digits). In the first case the letter is then reduced from list DL and added to list UL and the selected digit is added to list UD. The parameters NDL, NUL, and NUD represent the new lists for DL, UL, and UD after complete conversion.

1. Add a fact for predicate *gendigits* if list L is empty.

2. Add a rule for predicate *gendigits* to handle the case when the first letter from list L is not in list UL. Use the system predicate *member* for non-deterministic selection (backtracking) of a letter from list DL. Use the predicate *del1elem* of exercise 5.2 for deleting this selected digit from list DL (you can download an implementation form the web page too). Convert the next letters in recursive way.

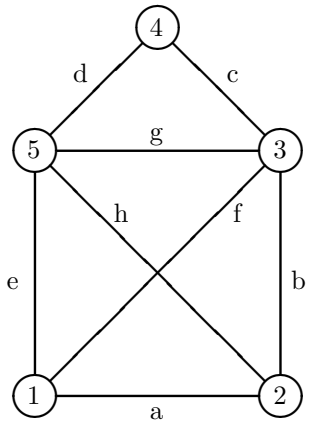
3. Add a rule for predicate *gendigits* for the case when the first letter from list L is already in list UL. Convert this letter using the predicate *convert* of exercise 5.3 (alternatively, you find an implementation to download from the web page). Convert the next letters in a recursive way.

4. Add a rule for predicate *cryptadd* which starts the conversion process for $S1$ using the predicate *gendigits* with the list of digits $DL = [0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9]$ and empty lists for UL and UD. After converting $S2$ and R with new, actualized lists for DL, UL, and UD, translate the list of digits into numbers using the predicate *horner* of exercise 3.5 (you may download an implementation from the web page) with the fixed value $X = 10$ and then check the correctness of the addition of the crypto-arithmetic puzzle.

b) Define, in analogy to exercise 6.1 a), a predicate *cryptmult*($S1$, $S2$, R, $D1$, $D2$, $D3$) which solves the corresponding crypto-arithmetic puzzle for multiplication of two numbers.

Example: $? - cryptmult([a, b], [a, b], [b, c, b], D1, D2, D3)$. yields
$D1 = [2, 6]$,
$D2 = [2, 6]$, and
$D3 = [6, 7, 6]$.

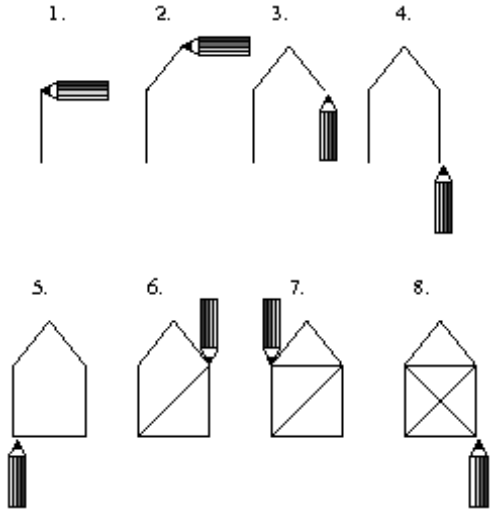
Exercise 6.2



"Das - ist - das - Haus - vom - Ni-ko-laus!" (or, "This is the house of Santa Claus!") are the eight syllables spoken while drawing segments in a sequence, ending up in the drawing of a house. The rules are:

- the pen must not be lifted,
- no edge is to be used more than once.

A possible correct solution is illustrated below.



How many different solutions exist?

Hints for implementation in Prolog

Start with specifying the facts which describe the topology of the house (knowledge base), i.e. by saying which nodes (vertices) are connected by which line segment (edge) using the predicate $line(Edge, Vertex1, Vertex2)$ (e.g. $line(a, 1, 2)$.). Be aware of symmetry!